Projekt zaliczeniowy - Sieci Komputerowe 2 laboratorium

Michał Boroń
mboron@cs.put.poznan.pl
www.cs.put.poznan.pl/mboron

Podstawowe zasady:

- Wszystkie projekty należy realizować w architekturze klient-serwer z użyciem protokołu TCP, chyba że uzgodniono inaczej z prowadzącym.
- Implementacje serwerów **współbieżnych** należy wykonać dla systemów operacyjnych GNU/Linux z użyciem języka C/C++ wykorzystując API BSD sockets.
- Implementację klienta można wykonać na dowolnej platformie z użyciem dowolnego języka programowania, po uzyskaniu zgody prowadzącego.
- Klient musi posiadać interfejs graficzny.
- Kody projektów są utrzymywane na repozytorium git w systemie GitLab https://gitlab.cs.put.poznan.pl Należy dodać prowadzącego do grupy projektu w systemie GitLab z uprawnieniami "Developer".
- Programy kompilują się bez błędów i ostrzeżeń (z flagą -Wall).
- Elementem zaliczenia jest sprawozdanie (maksymalnie 2 strony A4) w formacie PDF, które zawiera: opis protokołu komunikacyjnego, opis implementacji (struktury projektu), opis sposobu kompilacji i uruchomienia projektu.

Projekty nie będą oceniane lub ocena zostanie obniżona, w przypadku:

- implementacji serwera, która nie spełnia przynajmniej jednego z wymagań: współbieżna, napisana w C/C++, używa API bsd-sockets, kompilowana w systemie Linux,
- braku sprawozdania,
- niemożliwości prezentacji działania projektu w laboratorium lub przy użyciu własnego sprzętu,
- rażącej niezgodności funkcjonalności z uzgodnionymi wymaganiami,
- rażącego zaniedbania czytelności kodu,
- uzasadnionych wątpliwości dotyczących samodzielności pracy,
- innych odstępstw od ustalonych zasad.

Kryteria oceny projektu zaliczeniowego (maks. 100pkt)

Wybór tematu (przesłanie mailem skonkretyzowanego tematu oraz wybranych języków programowania)	0-5/0-2 (do 31.10 / do 30.11)
Utrzymywanie projektu na repozytorium git	0/2/5 (brak/kilka commitów/realne wykorzystanie)
Sprawozdanie (opis protokołu, struktury projektu, sposobu uruchomienia)	0-5 (jasność przekazu, zawieranie wszystkich elementów)
Protokół komunikacyjny (poprawność i adekwatność)	0-10
Zgodność funkcjonalności z uzgodnionymi wymaganiami	0-10
Brak podstawowych błędów (wbite na stałe adresy, localhost, itd.)	0-5
Poprawna kolejność wywołania funkcji sieciowych, możliwość podania nazw domenowych, obsługa błędów funkcji sieciowych, itd.	0-15
Działanie programu w niesprzyjającym środowisku - sklejanie wiadomości / fragmentacja przesłanych danych	0-10
Poprawność aspektów związanych ze współbieżnością	0-10
Interfejs graficzny - nie "zawiesza się", umożliwia podanie adresu/nazwy domenowej serwera, itd.	0-10
Brak błędów i ostrzeżeń przy kompilacji (z flagą -Wall)	0-5
Dodatkowe czynniki (łącznie **maks 10pkt**) = floor(suma składowych, 10)	
Look&feel / UX	0-3
Czytelność kodu, komentarze	0-3
Architektura programu (podział na struktury, klasy, funkcje)	0-3
Subiektywna ocena jakości projektu	0-3
Projekt oddany w pierwszym terminie	0-2

Oceny:

od 90p.	5.0
od 80p.	4.5
od 70p.	4.0
od 60p.	3.5
od 50p.	3.0

Przykładowe tematy projektów (można zaproponować własne)

Jednoosobowe:

1. Sieciowa turowa gra logiczna, np.: reversi, szachy, warcaby

   Jeden serwer do którego podłączają się gracze. Wspiera wiele równoległych rozgrywek między parami graczy.

2. Zdalne zamykanie systemów operacyjnych

   Jeden program-agent per węzeł z systemem operacyjnym do zamknięcia. Jeden serwer, który wie o wszystkich agentach oraz zna uprawnienia klientów.

3. Zdalna konsola - telnet

4. System współdzielonego kalendarza sieciowego

Dwuosobowe:

5. System komunikacji grupowej typu IRC (dość łatwe)

6. Komunikator internetowy typu GG (dość łatwe)

7. Komunikator internetowy typu Skype

8. System wymiany komunikatów publish/subscribe

9. Grupowy edytor plików tekstowych

10. Prosty serwer HTTP zgodny z RFC 2616 co najmniej w zakresie żądań GET, HEAD, PUT, DELETE

11. Prosty serwer FTP zgodny z RFC 959 co najmniej w zakresie komend: ascii, binary, mkdir, rmdir, put, get

12. E-mail: system poczty elektronicznej

13. System mnożenia dużych macierzy kwadratowych (lub innych obliczeń algebry liniowej)

14. System wymiany plików w architekturze peer-to-peer

15. Implementacja dowolnej usługi sieciowej zgodnie z RFC